N94-32428

A HYPERTEXT SYSTEM THAT LEARNS FROM USER FEEDBACK

Dr. Nathalie Mathé
NASA Ames Research Center[1]
Mail Stop 269-2
Moffett Field, CA 94035
mathe@ptolemy.arc.nasa.gov

ABSTRACT

Retrieving specific information from large amounts of documentation is not an easy task. It could be facilitated if information relevant in the current problem solving context could be automatically supplied to the user. As a first step towards this goal, we have developed an intelligent hypertext system called CID (Computer Integrated Documentation). Besides providing an hypertext interface for browsing large documents, the CID system automatically acquires and reuses the context in which previous searches were appropriate. This mechanism utilizes on-line user information requirements and relevance feedback either to reinforce current indexing in case of success or to generate new knowledge in case of failure. Thus, the user continually augments and refines the intelligence of the retrieval system. This allows the CID system to provide helpful responses, based on previous usage of the documentation, and to improve its performance over time. We successfully tested the CID system with users of the Space Station Freedom requirement documents. We are currently extending CID to other application domains (Space Shuttle operations documents, airplane maintenance manuals, and on-line training). We are also exploring potential commercialization of this technique.

INTRODUCTION

There is a need in NASA, and other governmental and commercial companies for rapid and effective access to information in large documentation systems. Accessing and using information stored in such documentation systems has become more and more difficult due to the increasing volume of documentation, the disparity of information sources, and the frequent rate of change of such documentation during its lifetime. One of the main problems in accessing information in large documentation systems is due to the lack of support for contextual information retrieval. The relevance of information not only depends on its contents, but also on the particular user and his/her current problem solving context.

Hypertext systems provide good support for browsing large amounts of documentation and for building associative links between various parts of the documentation. While hypertext systems increase accessibility of information, they do not provide any built-in selectivity mechanism. This increased accessibility may magnify an already severe problem of selection [1]. For these reasons, knowledge-based systems technology can be very helpful in alleviating the selection problem and the cognitive overhead of the user. Our approach on the CID project has been to develop an intelligent context-sensitive indexing and retrieval mechanism which interacts with and learns from the user [2]. This mechanism lets users filter information by context of relevance, and build personalized views of the documents over time. It also gives them the capability to share knowledge with other users.

RETRIEVING INFORMATION IN LARGE DOCUMENTS

We have analyzed uses of the Space Station Freedom (SSF) Program Requirement Documents. When looking for specific information, users usually employ both the table of contents (hierarchical browsing) and the index of the available documents to guide their search. Even then, however, the search is not very focused, and it is common for users to examine several places in the documentation before finding the information needed. Moreover, the needed information might be distributed in several locations, making it harder to exhaustively find all the relevant pieces. In other cases, the information might not be in the document at all, but the user has to be certain of it before requesting a change to the document.

---

[1] Dr. Mathé is currently working at NASA ARC under contract 21-1614-2360 to San Jose State University.

While in a small document the user could use a simple sequential trial-and-error decision process (go to each section mentioned in the index and read its text), this is not feasible in large documents. We observed that users prefer to avoid reading pages of text, and apply an iterative filtering strategy: select a rough set of candidates locations in the document, use high level information to decide which ones might be relevant, repeat this process until only a few locations have been identified, then access and search the text at each location. If the information is not found, or only partial information is found, then backtrack in the search by broadening the set of candidate locations. In such a strategy, reading text is used in the last resort.

To avoid repeating this lengthy process each time, users develop cognitive representations of the organization and content of the documentation. They build semantic descriptions of the locations already visited in the text, and contextual information around the descriptors already used to access particular pieces of information. These cognitive maps are thus context- and user-dependent. This memory however does not last very long unless the same information is retrieved often under the same context.

THE COMPUTER INTEGRATED DOCUMENTATION SYSTEM

Retrieving specific information from large amounts of documentation could be facilitated if information relevant in the current problem solving context could be automatically supplied to the user, in understandable terms and in a flexible manner. As a first step towards this goal, we have developed an intelligent hypertext system called CID (Computer Integrated Documentation) [3]. The CID system enables integration of various technical documents in a hypertext framework and includes an intelligent context-sensitive indexing and retrieval mechanism. The CID system has been used to help Space Station Level I personnel manage the Program Requirement Document for SSF. A typical screen of CID windows is presented in Figure 1. It contains the CID control panel that allows the user to control the entire library, and an example document. Both text and graphics capabilities are available within CID.

Structured Technical Documents

In existing technical documents, such as those common at NASA, information is structured hierarchically. Designing a complex system like the Space Station Freedom is an iterative process. Its documentation system is designed to handle a huge amount of information. It is organized around the Program Requirement Document (PRD), which establishes the highest level requirements associated with the Space Station Program. Generally, lower level documents expand upon the topics expressed in the PRD. For instance, Figure 1 shows the CID version of the PDRD (Program Definitions and Requirements Document, one level below the PRD) Section 3 (Systems Requirements), which contains about 900 pages, and uses 1.4 Mb of memory (without figures).

Each document includes several major nodes (e.g., change and revision notices, table of contents, a body of text segmented into sections and subsections, tables, figures, various appendices.) There are references between sections, or links in the hypertext "language", which are linear (section to next section) and non linear (reference to a section other than the next one). There are references to other major nodes within a document and to other documents.

In CID, we call *descriptor* [2] any word, phrase, or piece of graphics which provides a meaningful "starting point" for a search in the documentation. A *referent* is the address of any part of text or graphics. In the current implementation, a descriptor is typically a word in the index, and a referent a document section. Initial hypertext links between descriptors and referents are automatically built by CID, and may be modified by the user. CID also lets users index referents with concepts that are not necessarily words or term-phrases included in the text.

[2] The concept of descriptor is very important in information retrieval [4]. Building such descriptors requires expertise in the domain of investigation. We are using a technique developed by Mark Zimmerman [5] that allows full-text extraction of words associated with their frequency in the text.

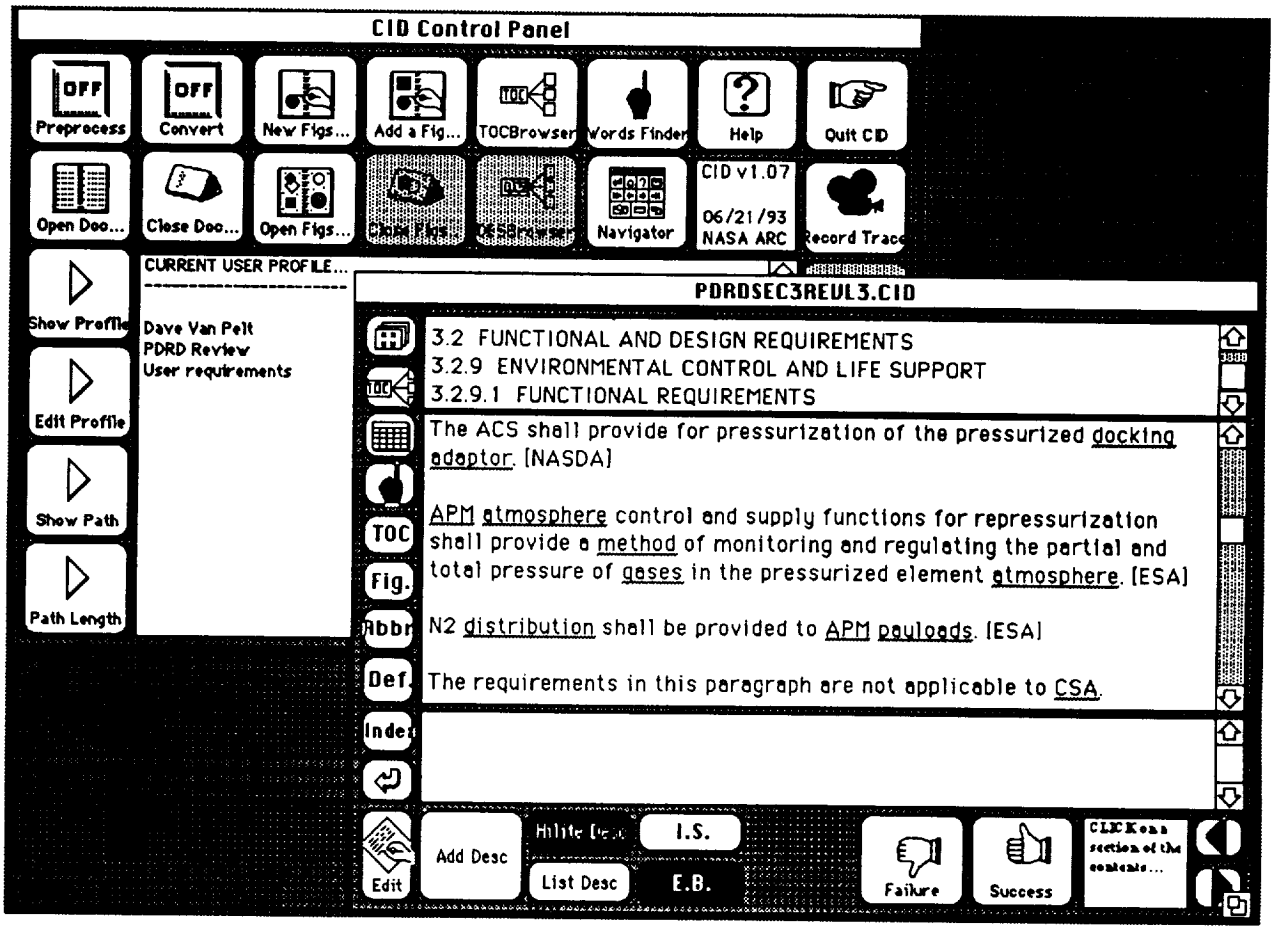


Figure 1 Part of a typical CID screen. The upper-left window is the CID control panel. The lower-right window is a CID document. The upper field includes the hierarchy of the displayed referent, the middle field is the actual text and the bottom field includes the next referents in the hierarchy. Clicking on the Success or Failure button automatically reinforces the displayed referent under the current user profile.

Context-Sensitive Retrieval

Contextual information characterizes the user, the types of task he/she is usually performing, and the type of information he/she is looking for. For instance, let us assume that one wants to retrieve some very specific information on the disposal of waste water in the Space Station. The first thing one may try is to browse the documentation with the descriptor "waste water." If the retrieval context can be specified, e.g., "You are a future scientific user of Space Station, and you want to know how experimental waste water will be handled," then a more efficient search can be accomplished. The set of relevant referents will not be the same under another context, e.g., "You are an engineer and you want to know how waste water will be recycled in the Environmental Control and Life Support system." Currently, in CID, the retrieval context is set by default on user log-on to one of the user's profiles, and can be modified by the user at any time.

Retrieving information in documentation is generally handled using keyword search. People find this very difficult in practice because keywords are used in a full-text search mode. Consequently, systems using keyword search come up with either hundreds of references or nothing [4]. To overcome this selection problem, CID uses the current retrieval context to filter down the number of referents for a descriptor that a user has to look at and thus narrows down the search. When the user selects a descriptor in CID, a list of ordered referents pops up. These referents have been found successfully in the *same or similar context* in past retrievals (the user can, however, access all the existing referents at any time). The order of referents is based on the past success and failure rates of each referent in this context. This is illustrated in Figure 2: the number of referents retrieved using the context filtering capability is narrowed down from 26 (total number of existing referents) to 2.

Users have the choice between filtering the list of retrieved referents based on previous reinforcements they have done themselves, or based on reinforcements done by someone with a profile similar to their own current profile. This second filtering option lets users with similar interests share their knowledge about the relevance of various pieces of information in a document. In the same way, a novice user can immediately visualize reinforcements done by a domain expert on the document, and shorten his/her training time.

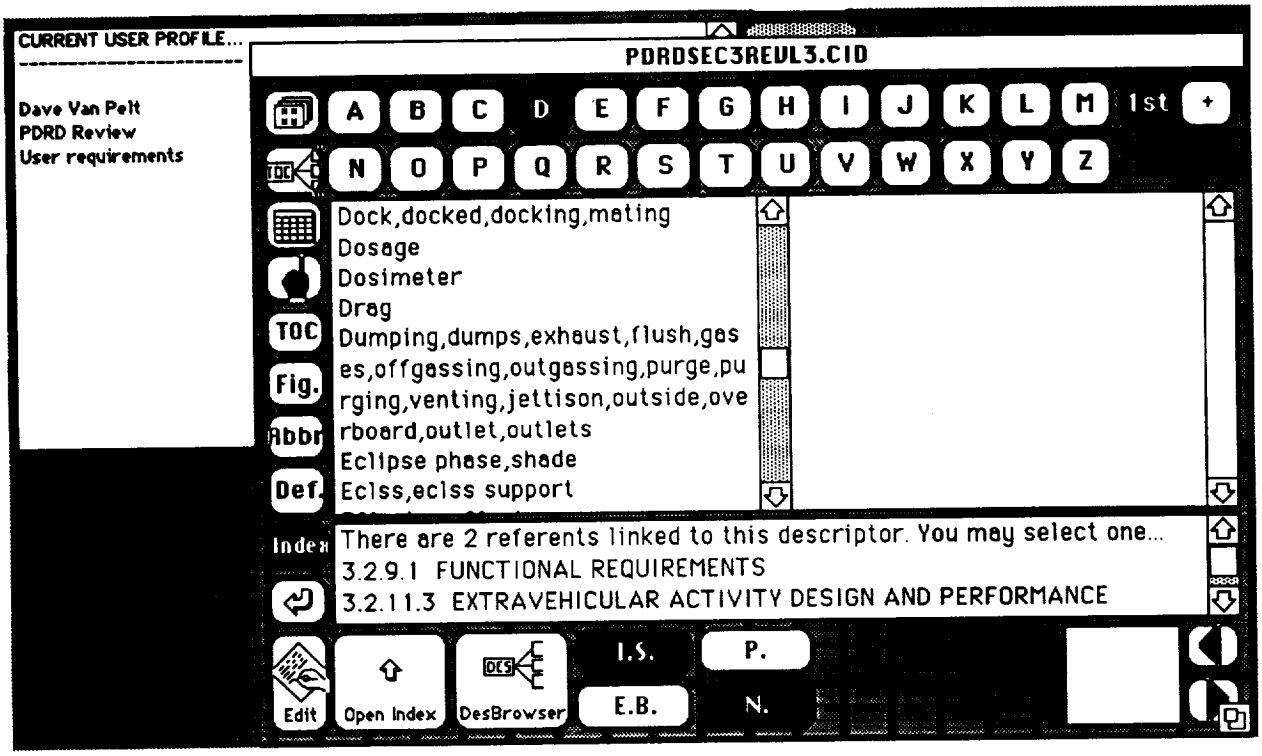


Figure 2 Use of an index in context. Here, the context was characterized by the current user profile. The user clicked on the descriptor "Dock". CID suggested two referents previously reinforced under this context.

Context-Sensitive Reinforcement

In order to narrow down the search using the current retrieval context, CID automatically acquires contextual knowledge from previous searches. This contextual knowledge is associated to the indexing links between descriptors and referents. CID modifies existing relations between descriptors and referents by using on-line user feedback to either reinforce or correct the system's knowledge in case of success or failure. This feature allows the system to tailor itself to the user.

After accessing a referent from a particular descriptor, the user can select either "Success" or "Failure" (cf. Fig. 1) to specify the relevance of this referent to his/her search. Using the current retrieval context, CID automatically updates the contextual knowledge attached to this link, updating contextual conditions and their respective weights (cf. Fig. 3). When a failure occurs, the system attempts to obtain from the user the reason for this failure. This learning mechanism enables CID to refine the indices over time to reflect their context of use.

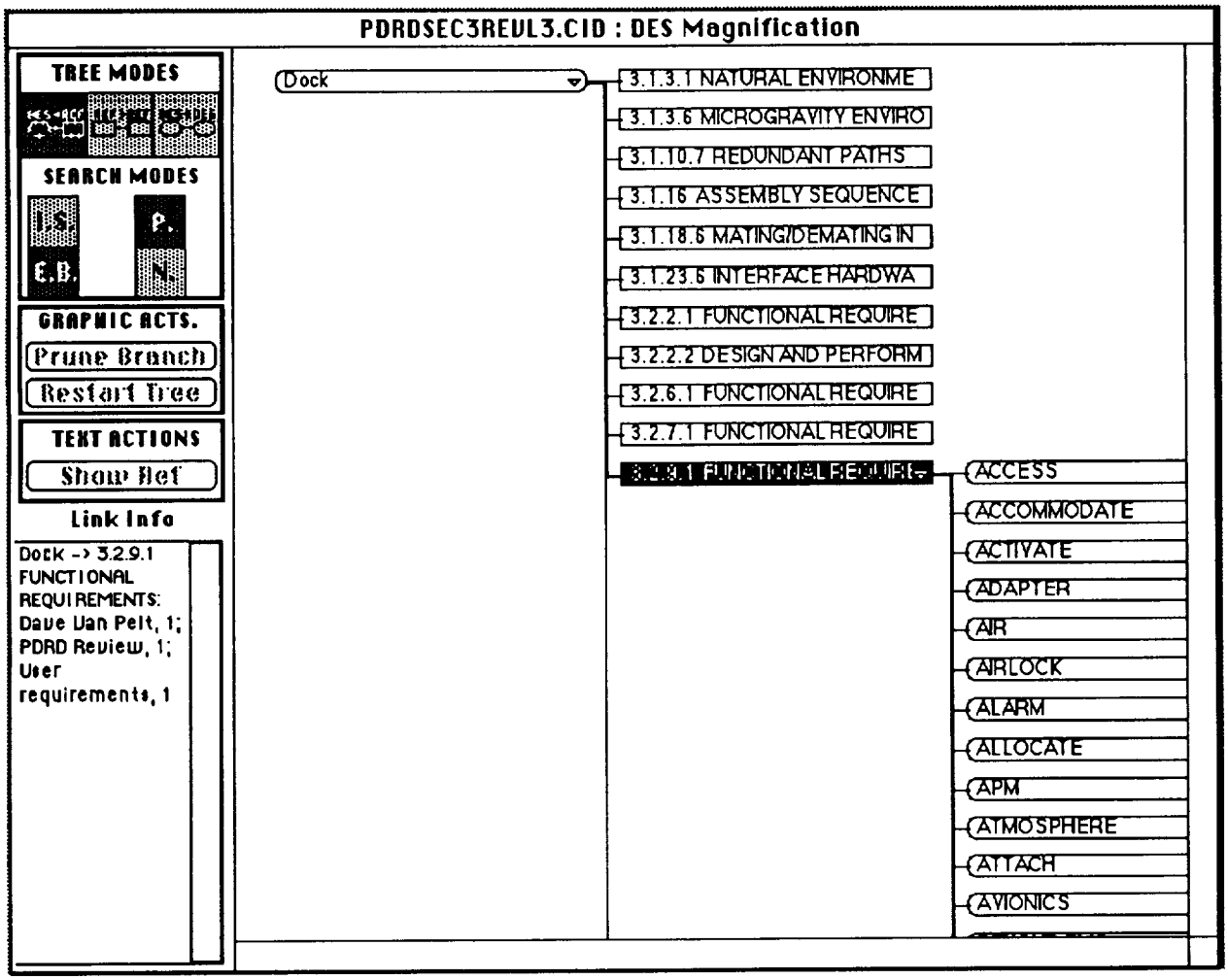


Figure 3 The graphical descriptor browser lets users visualize reinforcements (bold lines) and browse indexing links. Descriptors are displayed as rounded nodes, and referents as rectangular nodes. Contextual conditions associated to the selected link are displayed in the lower left corner. Clicking on a referent node displays the text of this section in the CID document.

CID Graphical Browsers

We have found that explicit maps of the documentation are very useful for quickly accessing information. These maps can be local ("where to go next?"), or global ("where am I?"). They can also present either the hierarchical structure of the documentation (local or global tables of contents), or the conceptual relationships between referents via descriptors (local or global conceptual indexes). In CID, these maps are embedded directly in the hypertext or accessible through graphical browsers.

Each section in a document locally displays information about its parent and children sections, and users can directly visualize in the text the words under which this section is indexed (cf. Fig. 1). These underlined words are active, and let users jump to other sections indexed under the same words.

User also have access to global maps displayed in graphical browsers. The graphical descriptor browser shown in Figure 3 lets users visualize reinforcements (bold lines) and browse relationships between referents and descriptors. In addition to browsing indexing links, they can ask for all referents similar to a given referent (sharing some descriptor with it), and filter the resulting list by context. The graphical table of contents browser lets users visualize the hierarchical structure of a document (cf. Fig. 4). In both browsers, all nodes are expandable and collapsible, and any section can be displayed directly in the corresponding document by clicking on a referent node.

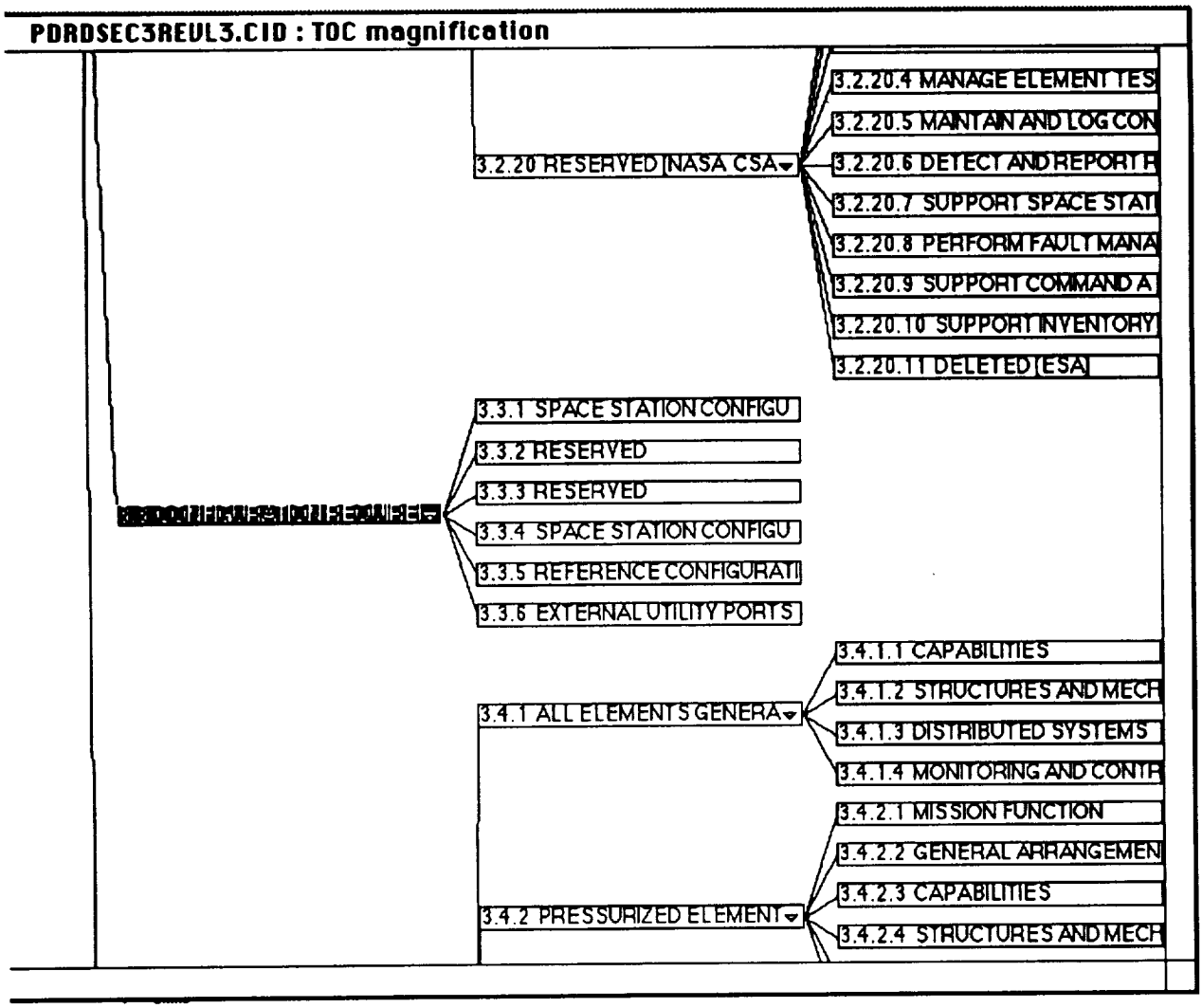


Figure 4 The graphical Table of Contents browser lets users visualize the structure of a document. Nodes can be expanded and collapsed. Clicking on a node displays the text of this section in the CID document.

USER TESTING RESULTS

In order to evaluate the effectiveness of and user satisfaction with CID learning capabilities, we conducted a set of user testing experiments [6]. Two domain experts from Space Station Level I personnel at NASA HQ used CID to find information in the Space Station Program Definition Requirements Document (PDRD). Their main task is to answer queries from future scientific users of the Space Station, regarding whether the current requirements fit the scientists' experiments needs. We chose 15 queries received by these experts from scientists, and not answered yet. Users were trained the previous week on CID for about half an hour. During each test session, we automatically recorded all their actions in CID, and asked them to verbalize what they were doing (think aloud protocol).

Overall, users found CID very user-friendly and were very eager to use the relevance feedback mechanism. In fact, they used it almost systematically each time they found some information they were looking for. With paper documents, they usually put yellow stickers and highlight the text with a yellow marker, in order to later on compose a report from the various pieces of information found in the document. With CID electronic documents, users only need to set up their user profile once (containing information to be used to personalize their feedback), then to click on the Success or Failure buttons whenever they want to give relevance feedback. Personalizing relevant information with their user profile also lets them share information with other users.

Apart from these positive results, users also requested more capabilities to make a better use of the CID system, including: better visualization of reinforcements directly in the document (highlighted sections, index terms, etc.); being able to reinforce any references, and not only these accessed from the index search; better filtering and access to previous reinforcements; being able to save reinforcements done for a query (which may involve several index searches or other types of searches) under one label; and means for transferring reinforcements done on a document to new revisions.

We are currently redesigning CID reinforcement mechanism in order to satisfy these new requirements. In this new version, users can reinforce any type of hypertext link, and have access to a broader type of queries, e.g., "show me all reinforcements I have done", "show me all reinforcements John has done for this query", "show me all reinforcements done yesterday", etc.

CONCLUSION

Besides providing an hypertext interface for browsing large documents, the ability of our system to automatically acquire and reuse the context in which previous searches were appropriate is unique. The design of contextual links to retrieve information is based not only on the way the documentation has been built, but also on user's information requirements and feedback when they are using the system. Thus, the user continually augments and refines the intelligence of the retrieval system. Context-sensitive information retrieval gives extended possibilities such as providing search expertise from other users, e.g., "what would John Smith do in this situation?"

We have shown that tailoring of hypertext documents during usage has been very well received by users. The main advantage is that it let them incorporate their knowledge and understanding of the content of a document over time, without disturbing the task they have to accomplish. This mechanism is extremely useful for large documents used by a large number of users, where the need for filtering information and building personalized views of the documents over time is important, as well as the possibility to share knowledge between users (access tailoring done by someone else, or a group of users).

Besides improving CID reinforcement mechanism as described in the previous section, we are extending CID to support operations at the Consolidated Control Center for Space Shuttle and Space Station in collaboration with the NASA Johnson Space Center. We are also studying the usefulness of CID for airplane maintenance electronic performance support with on-line documentation and training. Finally, we are exploring the potential commercialization of CID learning technique with computer software companies.

The current version of CID is implemented on a Macintosh in HyperCard and in C language. The CID software package is available for distribution to US universities and companies.

REFERENCES

1. Jones, W.P., "How do we distinguish the hyper from the hype in non-linear text ?," in *Proc. 1987 INTERACT'87.*. Holland: Elsevier Science, 1987.
2. Boy, G.A., "Acquiring and refining indices according to context," in *Proc. 1990 Fifth AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop.*. Banff, Canada: November 1990.
3. Mathé, N. and G.A. Boy, "The Computer Integrated Documentation Project: A Merge of Hypermedia and AI Techniques," in *Proc. 1992 Space Operations Application and Research (SOAR) Workshop.*. NASA Johnson Space Center, Houston, Texas: August 3-6, 1992.
4. Salton, G., *Automatic Text Processing: the transformation analysis, and retrieval of information by computers.* Redding, Ma: Addison Wesley. 1989.
5. Zimmerman, M., *TEXAS version 0.5*. Technical Report. Silver Spring, MD: 1988.
6. Mathé, N., "Tailoring Hypertext Documents in Context: First User Testing Results," Poster presented at the 1993 Hypertext conference. Seattle, Ca: Nov. 14-16, 1993.

COMMON MODELING SYSTEM FOR DIGITAL SIMULATION

Capt Rick Painter, USAF
USAF Wright Laboratory/Avionics Directorate
Wright Patterson AFB OH 45433

ABSTRACT

The Joint Modeling and Simulation System is a tri-service investigation into a common modeling framework for the development of digital models. The basis for the success of this framework is a X-window-based, open systems architecture, object-based/oriented methodology, standard interface approach to digital model construction, configuration, execution, and post processing.

For years Department of Defense (DoD) agencies have produced various weapon systems/technologies, and typically digital representations of those. These digital representations (models) have also been developed for other reasons such as studies and analysis, Cost Effectiveness Analysis (COEA) tradeoffs, etc.

Unfortunately, there have been no Modeling and Simulation (M&S) standards, guidelines, or efforts towards commonality in DoD M&S. The typical scenario is an organization hires a contractor to build hardware, and in doing so a digital model may be constructed. Until recently, this model was not even obtained by the organization. Even if it was procured, it was on a unique platform, in a unique language, with unique interfaces, and, with the result being UNIQUE maintenance required. Additionally, the constructors of the model expended MORE effort in writing the "infrastructure" of the model/simulation (e.g. user interface, database/database management system, data journalizing/archiving, graphical presentations, environment characteristics, other components in the simulation, etc.) than in producing the model of the desired system. Other side effects include: duplication of efforts; varying assumptions; lack of credibility/validation, decentralization both in policy AND execution, and various others. J-MASS provides the infrastructure, standards, toolset, and architecture to permit M&S developers and analysts to concentrate on the their area of interest.

J-MASS ARCHITECTURE and STANDARDS LAYERS

J-MASS has several architectural and standardization layers. This paper describes J-MASS in terms of the Tool Interconnect Backplane (IBP) layer, referred to as the Simulation Support Environment (SSE IBP) the Simulation Runtime Agent (SRA) IBP layer, and the Model Component/Object Standards layer.

MODEL COMPONENT/OBJECT STANDARDS

Each model component (or object) in J-MASS is structured compliant with our Software Structural Model (SSM). The SSM evolved from the Software Engineering Institute (SEI) work on the Object Connection Update (OCU) model. Both the C-17 and B-2 weapon systems trainers use a similar methodology for their object definition. The SSM, also described in a document, enforces software structure and interface standards for all levels of object decomposition. In this way, ANY objects in the system can be syntactically "connected" with any other objects in the system with guaranteed success. Semantically, the connection may have no realistic "meaning", but syntactically they can be connected ("Assembled", see discussion in Develop and Assemble Modes under Tool Interconnect Backplane). J-MASS objects are described in three layers: "Players", "Assemblies", and "Elements". Players are the "top" level objects responsible for synchronization with the simulation runtime engine and comply the software interface is standard to all objects at that level. Additionally, the interface between the "player", and its subcomponents, "assemblies" and "elements", is also standard. This interface is similar to but NOT exactly like the player to runtime engine interface. Figure 1 represents the J-MASS SSM implementation.

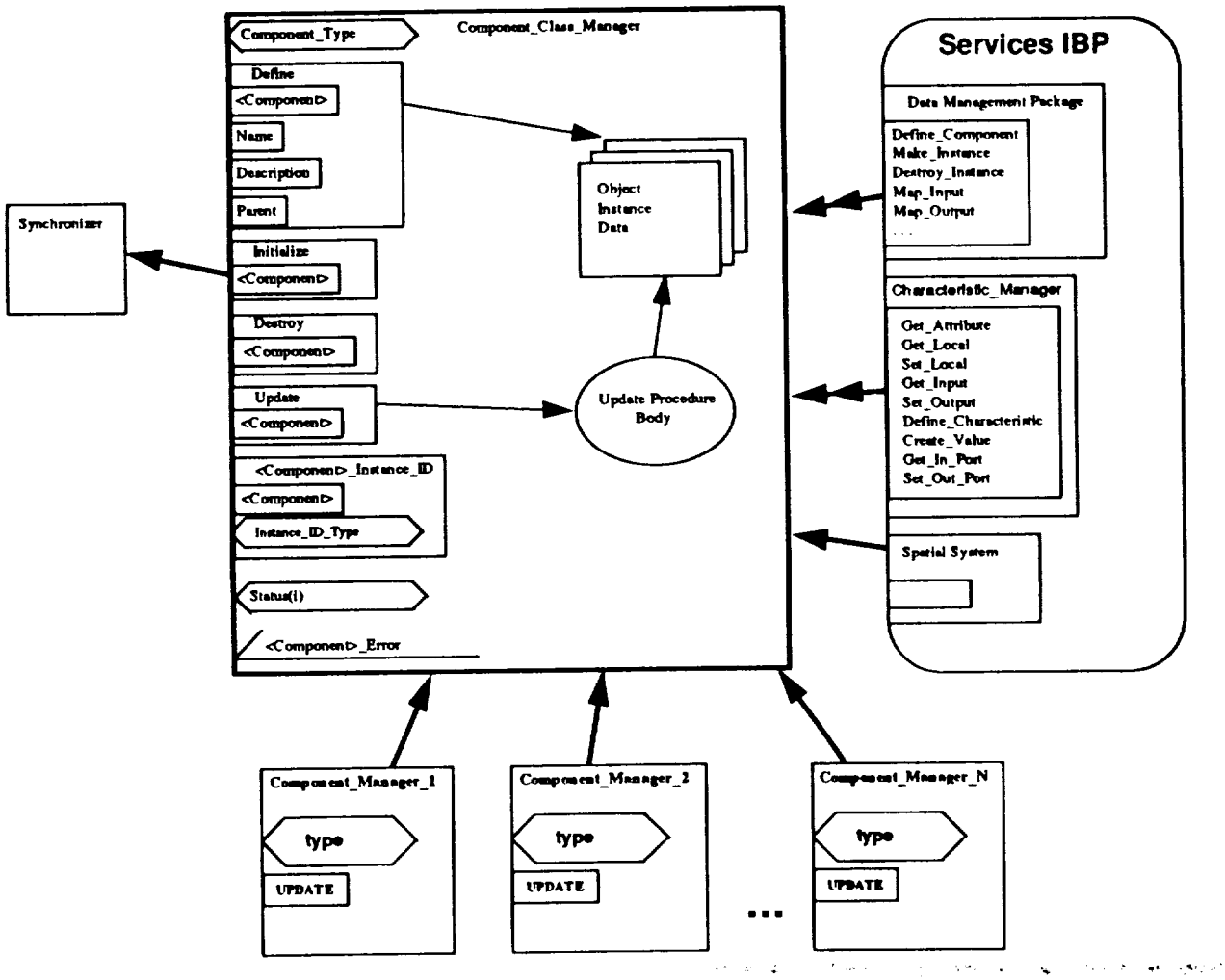
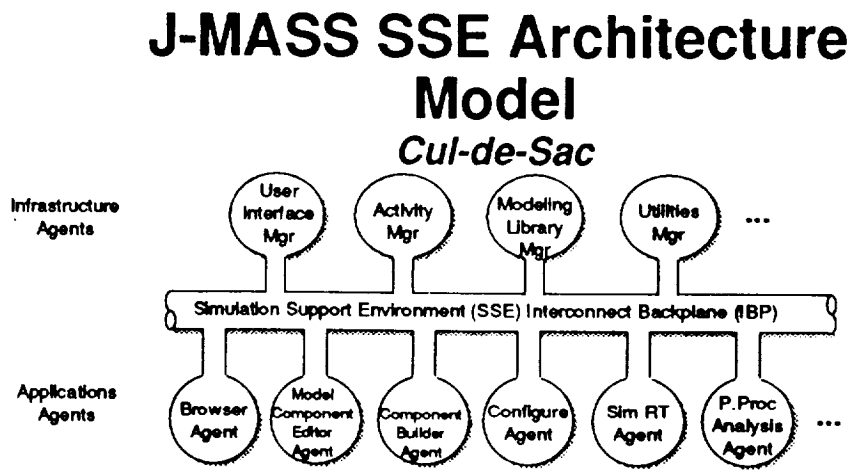


FIGURE 1

OPEN ARCHITECTURE - TOOL INTERCONNECT BACKPLANE

At the Tool Interconnect Backplane (IBP) layer, known in J-MASS as the Simulation Support Environment (SSE) IBP, several backplane methodologies were considered, including the HP Softbench, IEEE P-1175 "Toaster Model", the Atherton Backplane, and, significantly upon the Common Object Request Broker Architecture (CORBA) from the Object Management Group (OMG). In J-MASS terminology, a cul-du-sac model is employed, where each cul-du-sac represents a tool, or potentially a collection of tools or capabilities, referred to as "agents". Each "tool" or "agent" (a software capability), can register as a client/server with the backplane, indicating the service/message traffic of interest. The backplane maintains the knowledge of the other tools that have registered that can either provide the service, or will request the service. This concept is knows as message brokering and is powerful for de-coupling the tools from knowledge of other tools on the system. J-MASS has implemented a prototype of its design for this backplane in C on Unix workstations, currently SUN Sparc series, and Silicon graphics. Other platforms in progress include the IBM RS6000, and HP 9000 series, with DEC Alpha, and VAXstations in the plans. Reference Figure 2 for a graphical depiction of this concept.



- Infrastructure Agents
  - Agent With Specific Responsibilities
  - Always Installed And Available
- Application Agents
  - Connect Directly To SSE Interconnect
  - Register Own Services
  - Request Services To Be Performed
  - Are Loaded / Removed Dynamically
  - Communicate Via SSE Interconnect Message Language Grammar

FIGURE 2

User Modes.

J-MASS has five conceptual "user modes" associated with it. These are "functionally" oriented modes, namely: Develop; Assemble; Configure; Execute; and Post-Process. Each represents a capability that a model developer and/or simulation analyst requires to build, configure, execute, and analyze simulations. Each of these modes can be viewed as an instance of the cul-du-sac methodology. The next series of charts (2 thru 6) depict an instance of the backplane at the tool interconnect layer for each of the J-MASS modes.

Develop Mode/Assemble Mode.

Develop Mode and Assemble Mode provide the model developer with visual mechanisms for constructing model objects/object hierarchies, with data flows represented. Control flows (not currently implemented) will also be depicted so that model developers can separate control/activation of objects from data flow. The graphical information is then translated to ascii "dot" notation, referred to as .DSC (description) files. These .DSC files are then read by an automatic code generator, which generates source code compliant with the Software Structural Model (SSM) in various languages (currently Ada, C++). The SSM is discussed further in the Model Component/Object Standards section. At this point, the algorithms for the lowest level objects in the decomposition must still be described (currently, in the native language thru an editor). The code can then be compiled, linked, loaded and executed. A semantic tool, or "template" editor, is provided to build the semantic "template" information that describes "normal" assembly of the model components, which is done in "Assemble" Mode. Here in develop, the template semantics are generated. See Figures 3 and 4 for a graphical depiction of Develop Mode. Assemble mode permits the connection of the model objects built in Develop Mode visually. The "templates" are populated with actual object instance selections. All of the model components are stored in the modeling library, an object oriented storage mechanism which makes the information about the objects in J-MASS available to all other agents on the backplane.

Application Agents Supporting Develop

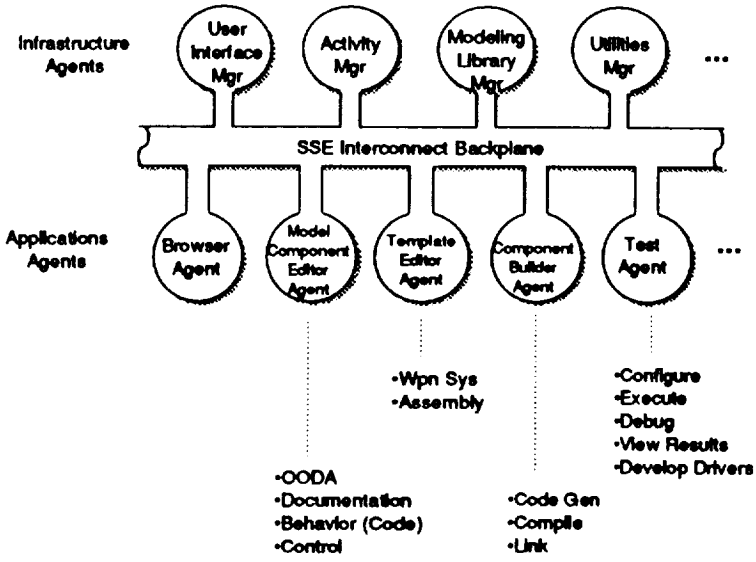


FIGURE 3

Application Agents Supporting Assemble

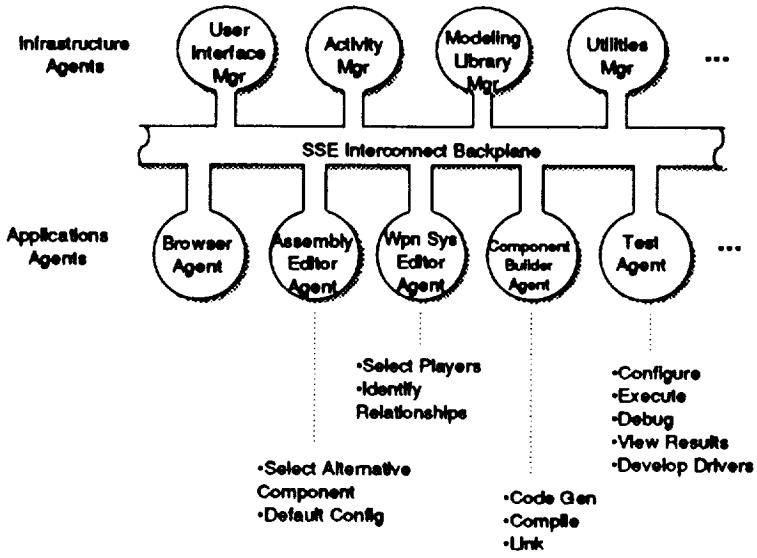


FIGURE 4

Configure Mode.

Configure Mode permits the M&S developer/analyst with the capability to determine simulation characteristics. Model component objects attribute values are populated with values thru a graphical configure tool. Additionally, geographical laydowns, raster maps, etc. are made available to set up the scenarios of the model objects stored in the model library. J-MASS "teams" are formed, whereby player classes are defined, and actual player instances are populated for the teams. This "distribution strategy" is totally configurable by the user. If "legacy" simulations exist, the configure mode will permit the modeler/analyst with the capability to catalogue those models/simulations, and have data passed back in forth sequentially. Eventually, real time synchronized communication between J-MASS compliant and legacy simulations will be achieved. Additionally, if a Distributive Interactive Simulation (DIS) Protocol Data Unit (PDU) generation is desired, the user is able to configure a J-MASS team (collection of players into a single executable). The entire team will then generate PDUs, and the J-MASS spatial system will create "objects" for the incoming DIS entities. The software that provides this capability is the DIS_manager software, and is de-coupled from the standard J-MASS objects, so as not to perturb that interface. Figure 5 depicts the architecture backplane instance for Configure Mode.

Application Agents Supporting Configure

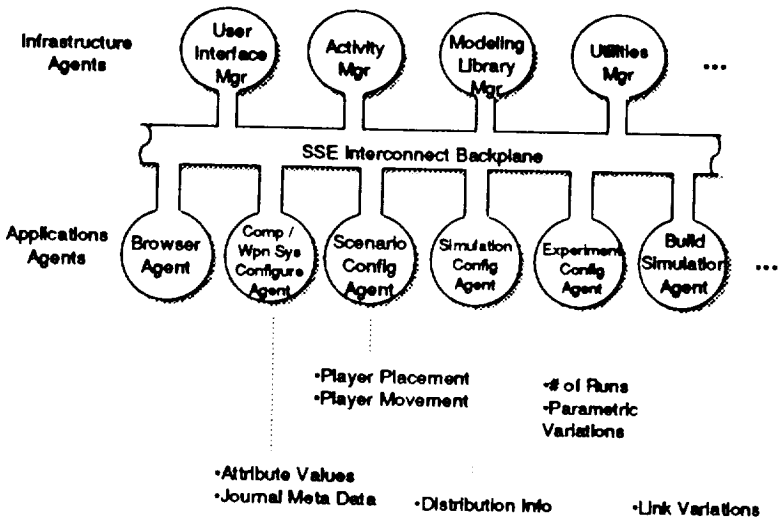


FIGURE 5

Execute Mode.

Execute Mode simply executes the selected simulation. Currently, visualization is accomplished in the Post Process Mode. If the DIS manager software was invoked due to configuration selection, then using "magic carpet" software, the PDUs can be displayed in real time. In work is a real time display of the simulation as it occurs. Figure 6 depicts the architecture instance for the Execute Mode.

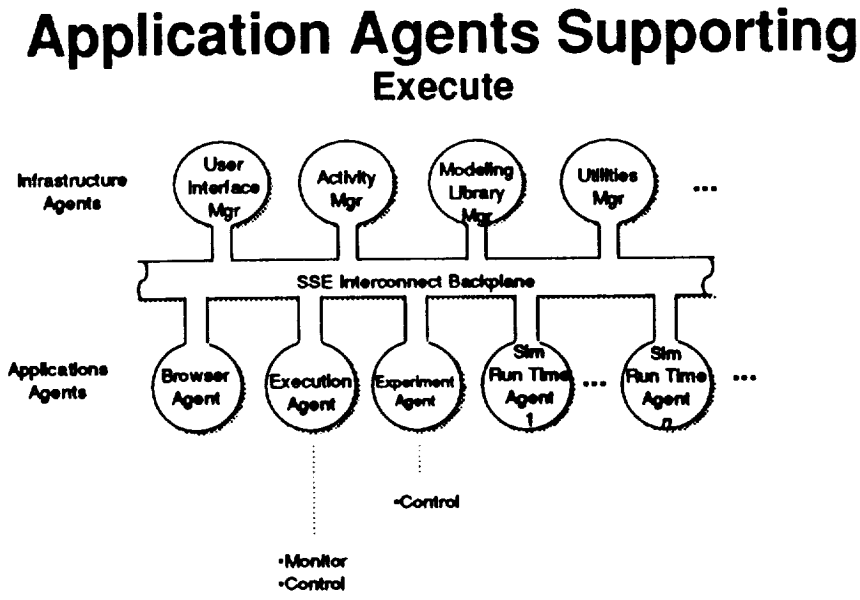


FIGURE 6

Post Process Mode

Post Process Mode is a visualization, both static and dynamic, of the information of interest to the user. This mode includes graphical plotting tools, and animated playback capability. The extraction tool converts the binary journalized data into ascii information. The filter mechanism then prepares it in the appropriate format for the display tool requested. Figure 7 describes the backplane instance for the post process mode of J-MASS.

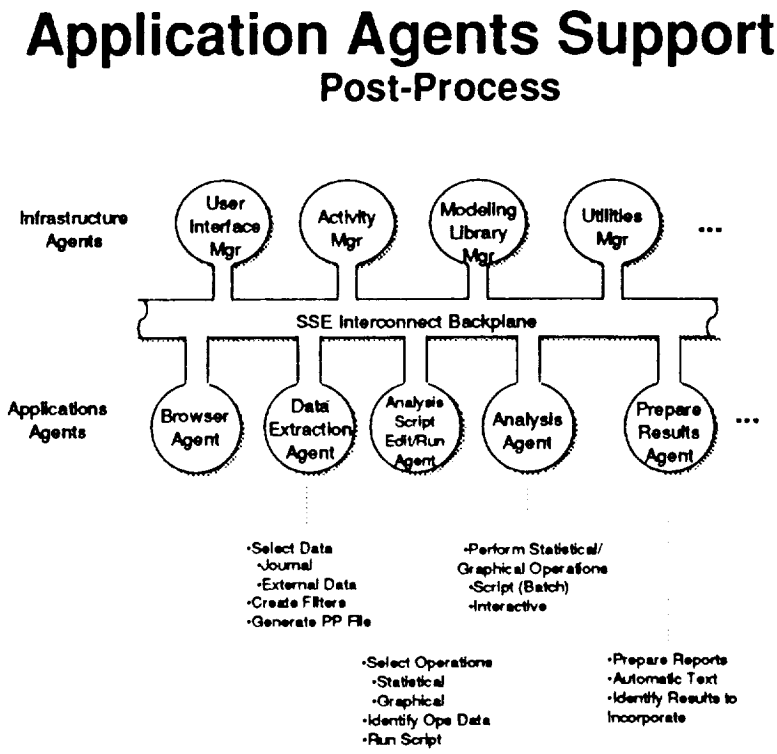


FIGURE 7

SIMULATION RUNTIME AGENT (SRA) ARCHITECTURE

The J-MASS Simulation Runtime Agent (SRA) architecture is depicted in Figure 6. The SRA is "expanded" in this view to show its own architecture. In fact, any agent on the SSE backplane may in fact be another recursive instance of the SSE level. Notice the SRA has its own backplane. The SSE level and SRA level backplanes could in fact be the same. In our current implementation, they are not, but both are distributed in nature using standard Unix (TCP/IP) socket message passing mechanisms. What is important to note in the SRA is the encapsulation of the spatial object, synchronization object, data management object, journalization object, and others away from the model objects. Thus, a true "plug and play" architecture is achieved because any given object may be replaced in the architecture without perturbing the other objects. In the SRA, each team is a single executable using a shared memory implementation, providing significantly faster communication than "inter-team" communication, which uses Unix sockets. Just as the SSE level architecture is distributable, so too are the "teams" within any given SRA. A J-MASS system may in fact have more than one SRA, each communicating over the SSE level backplane. In fact, we plan to demonstrate an Ada SRA with Ada model objects communicating with C++ SRA and C++ model objects over the SSE IBP mechanism. "Players" communicate with each other by placing information on each others "ports" facilities. Players do NOT require apriori knowledge of what team the other player is on, the team synchronizers work with the SRA synchronizer to "locate" the appropriate port. Again, the model objects remain "un-perturbed" with this approach. Journalization of output is accomplished by the journalization object, using state information maintained in the Data Management Package (DMP). In this way, non-intrusive journalizaing occurs. Figure 8 represents the expanded view of the SRA.

J-MASS Architecture
Simulation Runtime Agent
(SRA) Detail

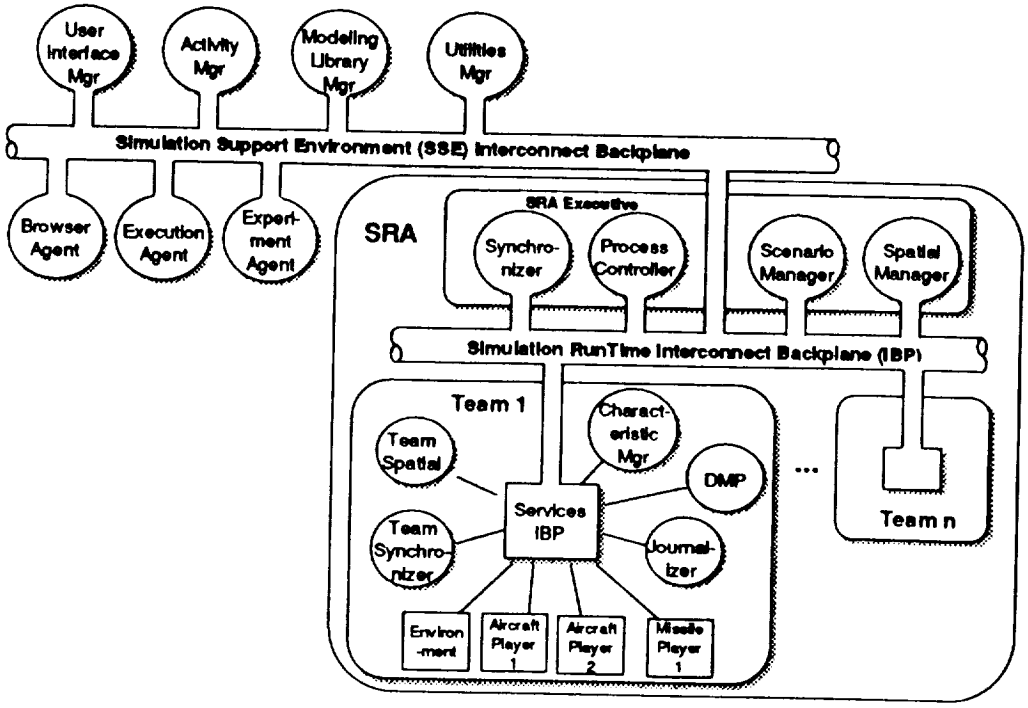


FIGURE 8

COMMERCIAL POTENTIAL

The J-MASS concepts and philosophies are not entirely original. The backplane methodology, message brokering mechanisms have been espoused by OMG and others. However, J-MASS has applied these concepts to a generalized Modeling and Simulation System.

J-MASS brings the idea of standards for digital simulations, both in structure and interface. This guarantees "plug and play" philosophies, both from model components and architecture components point of view. J-MASS espouses the idea of "plug and play" throughout the architecture to include tools, objects (model components), etc.

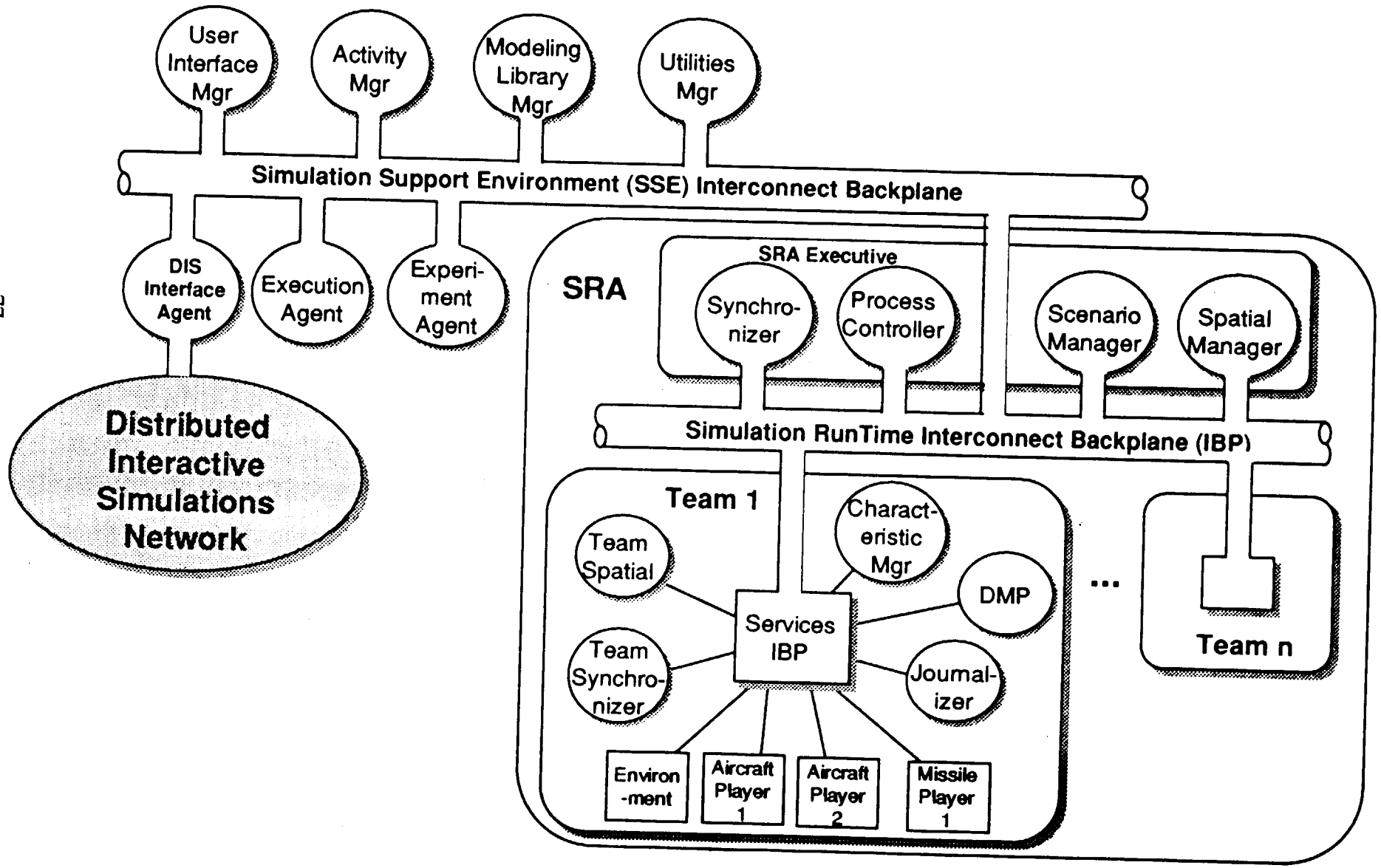
The J-MASS notion of graphical tool environment coincides with standard commercial technology as well. Expanding that concept which permits (automated) standard compliance with specified standard structures is another potential benefit to the commercial world.

J-MASS itself does NOT prescribe what objects or systems are modelled with its architecture. For example, the object repositories could represent traffic objects, manufacturing objects, weather objects, organizational objects, utility objects, etc. The system is designed so that the M&S communities build object hierarchies and behavior appropriate for the particular domain.

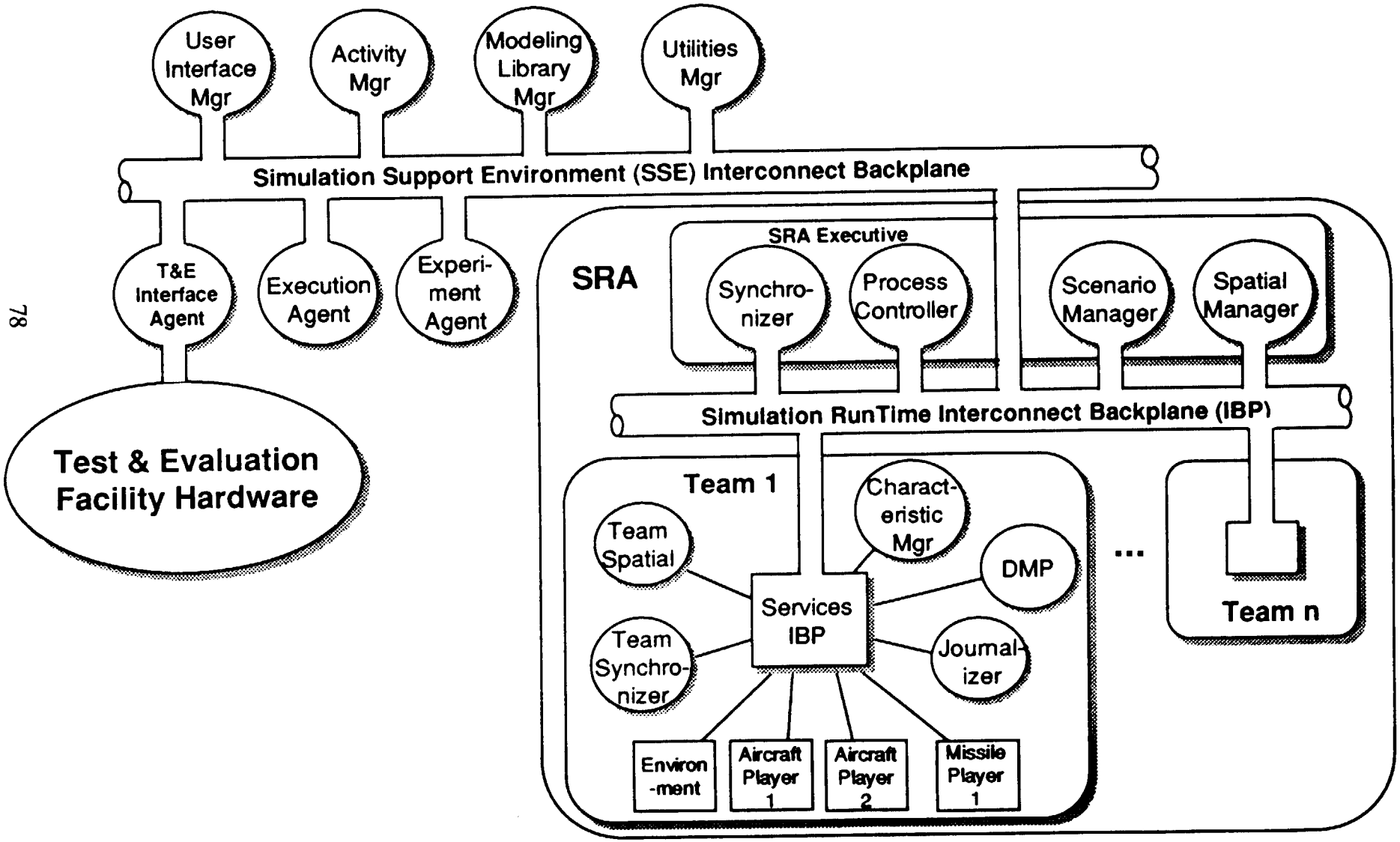
J-MASS / Windows Comparison

J-MASS	Windows
J-MASS Development Team	**Microsoft Corp.**
J-MASS Software	**Windows Operating System**
Tool Interconnect Standard	**Windows Interface Standard**
User Interface Mgr, Activity Mgr, Modeling Library Mgr, Utilities Mgr	Program Mgr, File Mgr, Print Mgr, etc.
Browse Agent, Test Agent, Simulation Runtime Agent (SRA)	Write, Paintbrush, Terminal, etc.
Specialized Post Processing Tools, Unique SRA, etc.	Word for Windows, Excel, PowerPoint, etc.
Model Interconnect Standard	**Word File Format**
J-MASS Model Component	Word Document

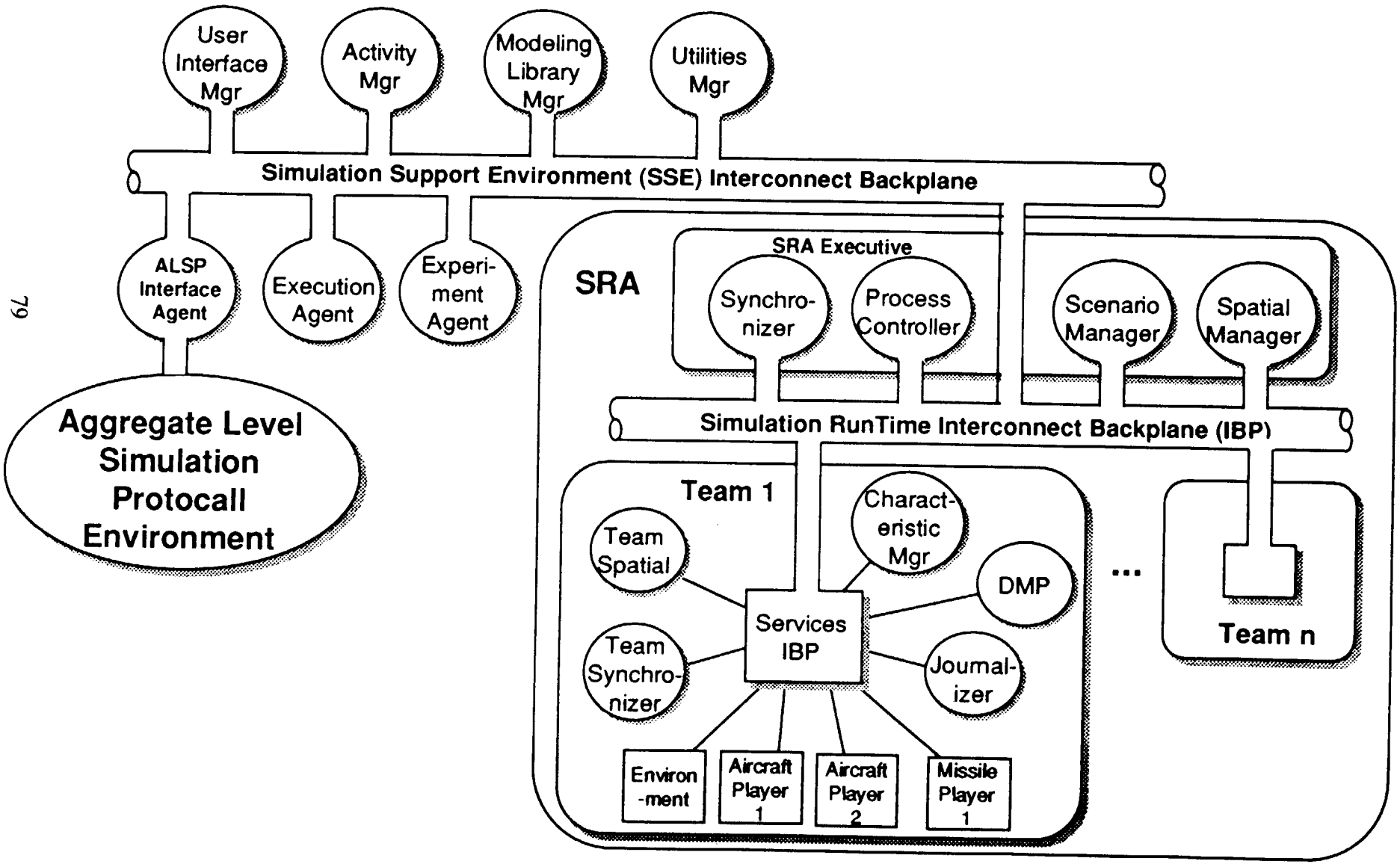
J-MASS Architecture
Extendible to Training



J-MASS Architecture
Extendible to Weapon Systems Acquisition



J-MASS Architecture
Extendible to Wargaming



ANALYTICAL DESIGN PACKAGE - ADP2 A COMPUTER AIDED ENGINEERING TOOL FOR AIRCRAFT TRANSPARENCY DESIGN

J.E. Wuerer, PDA Engineering
M. Gran, Wright Laboratory
T.W. Held, University of Dayton

ABSTRACT

The Analytical Design Package (ADP2) is being developed as a part of the Air Force Frameless Transparency Program (FTP). ADP2 is an integrated design tool consisting of existing analysis codes and Computer Aided Engineering (CAE) software. The objective of the ADP2 is to develop and confirm an integrated design methodology for frameless transparencies, related aircraft interfaces, and their corresponding tooling. The application of this methodology will generate high confidence for achieving a qualified part prior to mold fabrication.

ADP2 is a customized integration of analysis codes, CAE software and material databases. The primary CAE integration tool for the ADP2 is P3/PATRAN, a commercial-off-the-shelf (COTS) software tool. The open architecture of P3/PATRAN allows customized installations with different application modules for specific site requirements. Integration of material databases allows the engineer to select a material and those material properties are automatically called into the relevant analysis code. The ADP2 materials database will be composed of four independent schemas: CAE Design, Processing, Testing and Logistics Support.

The design of ADP2 places major emphasis on the seamless integration of CAE and analysis modules with a single intuitive graphical interface. This tool is being designed to serve and be used by an entire project team, i.e., analysts, designers, materials experts and managers. The final version of the software will be delivered to the Air Force in January, 1994. The Analytical Design Package (ADP2) will then be ready for transfer to industry. The package will be capable of a wide range of design and manufacturing applications.

1 INTRODUCTION

ADP2 is an integrated design tool consisting of existing analysis codes and Computer Aided Enginering (CAE) software. The objective of the ADP2 effort is to develop and confirm an integrated design methodology for frameless aircraft transparencies. ADP2 analysis capabilities include: aerodynamic heating, transient thermal response, static and dynamic structure response, optical ray trace, injection molding process simulation, and an aircraft transparency related material properties modeling and databank system. The design process is to be iterative and capable of producing frameless transparency designs, information needed for the design of integral aircraft interfaces, and information needed to support the design of injection molding tooling and the specification of molding process parameters for specific materials.

ADP2 is a second generation analysis system. The initial ADP development was initiated in 1989, References 1, 2 and 3. Since the inception of the original ADP, significant developments in both CAE and design support software relevant to aircraft transparency design have evolved. In addition, certain current design requirements, e.g., optics analysis and material properties modeling, were not addressed in the original ADP. Finally, significant advances in computing hardware have occurred making it possible to perform the required computations on workstation systems as opposed to mainframe platforms.

Recent developments in CAE design tools have introduced the ability to integrate special purpose and commercial-off-the-shelf (COTS) software in a user friendly (intuitive/interactive) environment. That is, to have a single user interface serve the primary analysis functions, specifically:

1. Modeling
   - Geometric Modeling (construction and modification)
   - CAD and IGES File Import
   - Graphics Manipulation